

Normalization: What and Why

When using a relational database management system the database is a collection of tables. A critical step in the database design process is deciding:

- How many tables do we need to store the data for an application?
- What data belongs in which table?

These questions have plagued database designers for many years. Initially we used intuition as our guide. We tended to group similar data items together in a table because it seemed to make sense to do so; for example all the data items describing an employee would be put in the employee table. Some designers seemed to have good intuition; they produced database designs that worked well resulting in applications that users were satisfied with. Other designers didn't have good intuition resulting in poor database designs and applications that users were not satisfied with.

The problem was there was no systematic method to help database designers decide how many tables there should be and what data items should be in which tables.

Codd developed a set of rules he called the rules of normalization, which we can apply when designing a RELATIONAL database. These rules guide us in deciding:

- The number of tables there should be in a database.
- The assignment of data items (attributes) to the various tables.

The rules of normalization have changed somewhat since Codd's initial version, but the normalization process remains the industry standard for designing a relational database.

A database design can be evaluated based on the following criteria:

- Does the design minimize redundant data?
- Does the design minimize the occurrence of anomalies within the data?

An anomaly is an inconsistency in the data stored in a database. There are three major types of anomalies: update, insert and delete.

We will look at how the rules of normalization produce superior database designs in terms of these criteria.

Minimizing Redundant Data

One criterion for a good database design is that the design minimizes redundant data. Consider the following example.

By intuition I could decide to create one table to store employee and department information as shown below.

EmpDept

<u>SSN</u>	Name	BDate	Dept no	DName	DMgrSSN
123456789	Smith, John B	09-Jan-55	5	Research	334455667
234567890	Wong, Franklin	08-Dec-45	5	Research	334455667
345678901	Zelaya, Alicia	19-Jul-58	4	Admin	223344556
456789012	Narayan, Remesh	15-Sep-62	4	Admin	223344556
567890123	English, Joyce	31-Jul-53	5	Research	334455667
678901234	Borg, James	10-Nov-27	5	Research	334455667
789012345	Jabbar, Ahmand	29-Mar-59	4	Admin	223344556
890123456	Wallace, Jennifer	20-jun-64	1	Marketing	445566778

Under this design the department name and the social security number of the department manager are repeated several times.

Applying the rules of normalization leads to the creation of two tables, one for the employee information and another for

the department information. Data redundancy is reduced (not eliminated) in the normalized design.

Employee

<u>SSN</u>	Name	BDate	Deptno (fk)
123456789	Smith, John B	09-Jan-55	5
234567890	Wong, Franklin	08-Dec-45	5
345678901	Zelaya, Alicia	19-Jul-58	4
456789012	Narayan, Remesh	15-Sep-62	4
567890123	English, Joyce	31-Jul-53	5
678901234	Borg, James	10-Nov-27	5
789012345	Jabbar, Ahmand	29-Mar-59	4
890123456	Wallace, Jennifer	20-Jun-64	1

Department

<u>Deptno</u>	DName	DMgrSSN
5	Research	334455667
4	Administration	223344556
1	Marketing	445566778

Minimizing Update Anomalies

An update anomaly occurs when we change the data in the database causing in an inconsistency in the data.

Suppose the manager of the research department retires and we hire a new manager. Under the intuitive design we must update four rows in the EmpDept table. If we update three correctly and forget to update the fourth row we have created an update anomaly in the database.

EmpDept

<u>SSN</u>	Name	BDate	Dept no	DName	DMgrSSN
123456789	Smith, John B	09-Jan-55	5	Research	222222222
234567890	Wong, Franklin	08-Dec-45	5	Research	222222222
345678901	Zelaya, Alicia	19-Jul-58	4	Admin	223344556
456789012	Narayan, Remesh	15-Sep-62	4	Admin	223344556
567890123	English, Joyce	31-Jul-53	5	Research	222222222
678901234	Borg, James	10-Nov-27	5	Research	334455667
789012345	Jabbar, Ahmand	29-Mar-59	4	Admin	223344556
890123456	Wallace, Jennifer	20-jun-64	1	Marketing	445566778

With the normalized design we need to update one row in the department table. No matter how we make the change we cannot get an anomaly in that all research department employees will have the same manager because the department is described by one row in the department table.

Employee

<u>SSN</u>	Name	BDate	Deptno (fk)
123456789	Smith, John B	09-Jan-55	5
234567890	Wong, Franklin	08-Dec-45	5
345678901	Zelaya, Alicia	19-Jul-58	4
456789012	Narayan, Remesh	15-Sep-62	4
567890123	English, Joyce	31-Jul-53	5
678901234	Borg, James	10-Nov-27	5
789012345	Jabbar, Ahmand	29-Mar-59	4
890123456	Wallace, Jennifer	20-Jun-64	1

Department

<u>Deptno</u>	DName	DMgrSSN
5	Research	22222222
4	Administration	223344556
1	Marketing	445566778

Minimizing Insert Anomalies

An insert anomaly occurs when we add new data to the database causing an inconsistency in the data.

Suppose we hire a new employee named John Doe. Under the intuitive design we must know all the information about the department the employee works for in order to add the information about the new employee to the database. If the department related information is added incorrectly we have created an insert anomaly in the database.

EmpDept

<u>SSN</u>	Name	BDate	Dept no	DName	DMgrSSN
111111111	Doe, John	17-May-70	5	Admin	334455667
123456789	Smith, John B	09-Jan-55	5	Research	334455667
234567890	Wong, Franklin	08-Dec-45	5	Research	334455667
345678901	Zelaya, Alicia	19-Jul-58	4	Admin	223344556
456789012	Narayan, Remesh	15-Sep-62	4	Admin	223344556
567890123	English, Joyce	31-Jul-53	5	Research	334455667
678901234	Borg, James	10-Nov-27	5	Research	334455667
789012345	Jabbar, Ahmand	29-Mar-59	4	Admin	223344556
890123456	Wallace, Jennifer	20-jun-64	1	Marketing	445566778

With the normalized design we do not need to know or enter all the department information in order to enter the new

employee information. This minimizes the likelihood of an insertion causing an anomaly.

Employee

<u>SSN</u>	Name	BDate	Deptno (fk)
111111111	Doe, John	17-May-70	5
123456789	Smith, John B	09-Jan-55	5
234567890	Wong, Franklin	08-Dec-45	5
345678901	Zelaya, Alicia	19-Jul-58	4
456789012	Narayan, Remesh	15-Sep-62	4
567890123	English, Joyce	31-Jul-53	5
678901234	Borg, James	10-Nov-27	5
789012345	Jabbar, Ahmand	29-Mar-59	4
890123456	Wallace, Jennifer	20-Jun-64	1

Department

<u>Deptno</u>	DName	DMgrSSN
5	Research	222222222
4	Administration	223344556
1	Marketing	445566778

Suppose the company expands and creates a new department. Under the intuitive design the new department information cannot be recorded in the database until the first employee is hired for the department (the employee social

security number is the primary key for the EmpDept table). With the normalized design new departments can be added any time because the department information is kept in its own table.

Minimizing Delete Anomalies

A delete anomaly occurs when we delete data from the database causing in an inconsistency in the data.

Suppose we fire all employees in the research department with the intent of hiring better employees to replace them in the near future. Under the intuitive design when the last employee for the research department is deleted we lose the information about the research department itself. Deleting employee information should not cause a loss of information about a department.

With the normalized design we are free to delete employee information without losing department information because each is stored in its own table.

By applying the rules of normalization to design a relational database we create a design that **minimizes** the amount of redundant data within the database and **minimizes** the likelihood of creating inconsistencies within the database during the life of the application.